Universidad
Carlos III de Madrid

**ÁLVARO GARCÍA - MOYA HERRERA**

# DESARROLLO DE SOFTWARE MÓVIL MULTIPLATAFORMA

Proyecto de Fin de Carrera - Resumen

Tutores:
Mikko Tiusanen
Juha-Matti Vanhatupa
Tutores y tema aprobados por el
Consejo de la Facultad de Ingeniería
Informática y Eléctrica
03.04.2013

# 1. INTRODUCCIÓN

La tecnología se ha convertido en una parte fundamental de nuestras vidas. Los ordenadores han sido aceptados como una método para educar a las nuevas generaciones. Los teléfonos móviles son más importantes para algunas personas que la cafeína o el ejercicio. De hecho, el número de dispositivos móviles en uso excede al número de personas con zapatos, de manera global.

Los juegos para móvil son, con diferencia, la razón más generalizada por la que la gente usa sus teléfonos móviles; superando incluso a la utilización del móvil para otras funciones tales como la comprobación del tiempo, los mapas o las redes sociales. Los juegos han crecido en complejidad y lo siguen haciendo según la tecnología avanza. Los juegos incluyen ya características como la inteligencia artificial o complejos motores de funcionamiento. Sin embargo, los juegos más exitosos son aquellos que incluyen solo unos pocos elementos de juego pero muy bien creados.

Este proyecto de fin de carrera considera el desarrollo de software móvil multiplataforma. Para ello, se va a crear un juego para móvil educacional orientado a la educación vial para niños. El juego se implementará en JavaScript, usando Appcelerator Titanium, como entorno de desarrollo integrado. Después, se evaluará el funcionamiento del juego sobre dos plataformas móviles: iOS 6 y Android OS 4.

# 2. DESARROLLO DE SOFTWARE MÓVIL MULTI-PLATAFORMA EFICIENTE

La implementación de software en dispositivos móviles es complicada, todo es más complejo de conseguir que lo es en web o en escritorio. Un debate existente entre desarrolladores es averiguar qué solución es mejor, entornos nativos o multiplataforma. Usando entornos nativos, las aplicaciones son rápidas y se tiene acceso completo a los recursos del dispositivo. Esas aplicaciones están disponibles en tiendas online, como Apple Store y Google Play. Sin embargo, aplicaciones de entornos nativos son costosas de desarrollar, en el sentido tanto económico como de tiempo, y además necesitan pasar un proceso de evaluación para ser publicadas.

La idea detrás de muchos sistemas multiplataforma es reducir el tiempo de desarrollo, siendo la solución ideal implementar un solo código y ejecutarlo en todas las plataformas, por tanto, desarrollar en entornos multiplataforma es más beneficioso. Los desarrolladores pueden ahorra tiempo codificando en un solo lenguaje de programación, que puede ser compilado y ejecutado en varias plataformas. No obstante, los entornos multiplataforma tienen otros problemas, como la necesidad de adaptarse a los diferentes tamaños de la pantalla o la falta de memoria para ejecutar aplicaciones. El aspecto más importante sobre aplicaciones multiplataforma es crear un único código que esté disponible para ejecutar en diferentes plataformas. Otro aspecto importante es investigar cuales son las diferencias funcionales del código en cada plataforma.

El enfoque para este problema es constructivo. Se va a presentar y evaluar la implementación de un juego educacional, que se ejecute usando un entorno multiplataforma. En la evaluación del proyecto se proporcionan los resultados de los test ejecutados sobre la aplicación. En estos resultados se puede comprobar las diferencias en la ejecución en diferentes plataformas.

# 3. TIWALKINGSAFE

La aplicación móvil desarrollada en este proyecto de fin de carrera es un juego para móvil orientado a la educación vial y sus peligros. Se llama TiWalkingSafe y se desarrolla en un mundo en dos dimensiones, 2D, para hacer lo más sencillo posible para el usuario aprender a interactuar con el juego. El juego no tiene límite de tiempo para realizar acciones o llegar a algún punto del mapa. La razón de que no esté basado en tiempo, es para enseñar a los usuarios que no deben tener prisa cuando se está cruzando la carretera en la vida real.

Cada nivel tiene una serie de “puntos de sincronización” y el objetivo es recogerlos. Cada punto de sincronización contiene información sobre un nuevo destino al que usuario tiene que llegar. Para poder coger estos puntos, el usuario tiene que llevar al personaje a través del mundo de la manera más segura que se pueda, evitando los peligros de la carretera. Cuando el usuario lleva al personaje delante de uno de estos puntos, y pulsa sobre el punto en la pantalla, el juego comprueba si el personaje está posicionado correctamente y de ser así, el usuario recibe nuevas instrucciones, pero si no es así, el juego avisa al usuario que debe llevar al personaje a una posición más cercana al punto. El juego está pensado para permitir a los usuarios interactuar con el propio juego, y aprender sobre educación vial, cómo funcionan los semáforos, y evitar situaciones peligrosas. Como en casi cualquier juego, hay una historia para motivar al usuario a jugar. En este caso, la historia es sobre un personaje de 8 o 9 años, que tiene que ir a varios sitios después del colegio, para hacer recados que su madre le ha mandado antes de volver a casa para cenar. Estos recados pueden ser: ir al supermercado, al entrenamiento de baloncesto o recoger a su hermana pequeña del parque y volver los dos a casa. Para poder hacer eso, el usuario controla al personaje a través del mapa diseñado, que contiene coches, semáforos y pasos de peatones.

El juego proporciona conocimiento sobre educación vial a los usuarios que invierten el suficiente tiempo jugando con él. Los usuarios aprenden cuando fallan, y sobretodo aprenden porqué han fallado y como podrían evitarlo la próxima vez.

# 4. IMPLEMENTACIÓN DEL JUEGO

La arquitectura del juego sigue el Modelo-Vista-Controlador. En dicha arquitectura la capa vista es la interfaz de usuario del juego, la capa del controlador se encarga de transformar las acciones del usuario en la interfaz en peticiones para el modelo, y finalmente la capa modelo controla todos los cambios producidos por el usuario. A través de Appcelerator Titanium, las tres capas se han implementado usando JavaScript como lenguaje de programación.

Los coches y semáforos son entidades independientes del juego, es decir, no están controlados por el usuario, sino que son dirigidos por la lógica del juego. Los coches están implementados para parar cuando el semáforo esté verde para los peatones, siendo rojo para ellos, y se muevan cuando el semáforo esté rojo. Los coches se desplazan por el lado derecho de la carretera, así como para el 66% de la población mundial. Los semáforos, por su lado, cambian su color, de rojo a verde y viceversa, cada cierto tiempo, expresados en segundos.

El usuario controla el personaje a través de una serie de botones en la parte baja de la pantalla. Este set de botones permite al usuario manejar al personaje, desplazándolo arriba, abajo, a la izquierda, a la derecha o incluso en diagonal. Cuando el usuario ha dirigido al personaje al punto de sincronización y pulsa la pantalla donde se encuentra el símbolo del punto, el juego comprueba que el personaje está correctamente posicionado, y de ser así el usuario recibe nueva información de a dónde dirigirse y tanto coches como semáforos paran de moverse, hasta empezar la nueva misión. Si no está correctamente posicionado, el juego le muestra al usuario un mensaje para mover al personaje a una posición más cercana al punto para recibir nueva información.

# 5. EVALUACIÓN

La evaluación ha tenido lugar en un entorno de emulador, ya que una evaluación en el dispositivo se quedó descartada debido al alto coste que supone. El autor realizó las pruebas y los resultados han sido expuestos en varias tablas. La tabla 5.1 ilustra la comparación final entre la puntuación obtenida en la evaluación de la aplicación sobre iOS and los resultados de la evaluación con Android OS.

**Tabla 5.1.** *Comparación entre los resultados obtenidos para ambas plataformas*

| Plataforma | Puntuación total |
|---|---|
| Apple iOS | 150.9 |
| Android OS | 136.8 |

Queda claro que la puntuación obtenida con iOS es ligeramente mayor que la puntuación de Android OS. La máxima puntuación que se podía conseguir en la evaluación es 205. En porcentajes, iOS tuvo un 74% de éxito, mientras que Android OS tuvo un 67% de éxito. En ese sentido, sólo hay un 7% de diferencia entre las dos evaluaciones, de manera que iOS ha funcionado un poco mejor que Android OS. A continuación se van a explicar las diferencias obtenidas en los resultados.

La primera diferencia en la evaluación es el tiempo de carga del juego. Android OS necesita más tiempo que iOS para cargar el juego. Esto se debe al emulador, ya que el emulador de Android es más pesado que el emulador de iOS. Para Android, implica mucho más tiempo porque crea funcionalidades y conexiones innecesarias, como hilos de actividad, analíticas o gestión de la batería. Por otra parte, cuando el juego comienza, el emulador de Android comprueba en cada movimiento, las características de todos los objetos del juego, como coches, semáforos o el personaje. Por el contrario, iOS no comprueba todos los objetos en cada movimiento, únicamente al comienzo del juego y al finalizar el mismo.

Otra diferencia es en la resolución de la pantalla de iOS y la de Android OS. La pantalla de Android es más grande que la de iOS, de manera que la resolución de los obje-

tos, como el personaje, los coches y los semáforos difiere. De hecho, los objetos aparecen desproporcionados, pero esto se podría corregir si las dimensiones de los objetos estuvieran expresadas en porcentajes, en lugar de en valores exactos. Existe un problema cuando se usan porcentajes, y es que el movimiento de los objetos se ve afectado, haciéndolo inexacto.

A pesar de que el emulador de Android OS es más lento que el de iOS, una vez cargado, Android OS proporciona servicio incluso si el usuario se dedica a pulsar repetidas veces los botones. Esta reacción en el usuario no es común, pero ocurre a menudo cuando los usuarios están muy concentrados jugando e intentan terminar el nivel lo más rápido posible, y pulsan los botones mucho más rápido de lo normal o en repetidas ocasiones. Este comportamiento no suele estar recogido en las evaluaciones de los juegos y el resultado suele ser que la aplicación se pare o se apague porque es incapaz de manejar todas las peticiones del usuario.

A diferencia de Android OS, iOS no ofrece movilidad al personaje si recibe estas secuencias de pulsaciones múltiples de los botones. El emulador de iOS tarda unos pocos segundos en reaccionar y seguir funcionando. Cuando se estuvo evaluando estas características, el personaje no se movía de su posición a ninguna nueva. Esto podría deberse a que iOS no almacena las peticiones para realizarlas más tarde. Probablemente recoja las nuevas y las gestione.

Las pruebas de aceptación e integración han conseguido la misma cantidad de puntos para ambas plataformas, ya que los casos de test están relacionados con la aplicación y sus requisitos. En ese sentido, es comprensible que la puntuación sea la misma para ambas plataformas, pero las pruebas de aceptación e integración son necesarias en cualquier proceso de evaluación de software. Finalmente, las pruebas de regresión han alcanzado resultados positivos para ambas plataformas. Es importante este hecho porque unos resultados bajos en la evaluación de regresión significan que cuando se corrige un error, aparecen más aún, de manera que el código nuevo es peor que el anterior.

# 6. CONCLUSIONES

Las aplicaciones multiplataforma son el siguiente paso en el mundo del desarrollo software móvil. Es la opción más barata, en términos de recursos y tiempo, para desarrolladores o pequeñas y medianas empresas que quieren ofrecer sus productos y servicios. Sin embargo, hay una desventaja en el desarrollo multiplataforma. Cuanto más grande se hace la aplicación más complicado es mantener la misma codificación para todas las plataformas, porque cada plataforma necesita diferentes implementaciones, es decir, cuesta más tiempo y recursos de los que ahorra. Si la aplicación es compleja, los desarrolladores necesitan encontrar nuevas maneras de codificar y usando entornos nativos no es necesario. Como se menciona anteriormente, este proyecto se considera cómo desarrollar software multiplataforma para móvil de manera eficiente. En ese sentido, la herramienta multiplataforma ofrece buenas características de desarrollo de software, ya que la aplicación fue diseñada para ser lo más sencilla posible.

Los juegos para móvil han crecido en complejidad y lo siguen haciendo conforme la tecnología avanza. Sin embargo, existen juegos muy buenos que sólo incluyen unas pocas funcionalidades, extremadamente bien implementadas. El juego educacional para móvil desarrollado en este proyecto es una solución razonable para educación vial. El juego es simple y pequeño, pero ofrece toda la funcionalidad básica para recrear la realidad de la carretera, y suficiente para evaluar cómo funciona y abstraer unos resultados para este proyecto. En caso de tener un juego más complejo, probablemente hubiera sido mejor opción usar entornos nativos, ya que al final los desarrolladores ahorran más tiempo que usando entornos multiplataforma, donde, el lenguaje de programación usado no siempre cumple las expectativas o soporta la funcionalidad requerida por el desarrollador.

La evaluación de la aplicación ha sido una tarea complicada. Además, la evaluación sería completa si se pudiera evaluar la aplicación en dispositivos reales, porque usando emuladores, siempre hay una pequeña posibilidad de que el resultado no sea completamente correcto. Con el emulador, iOS trabaja significativamente mejor que Android,

carga más rápido, y no realiza comprobaciones innecesarias. Sin embargo, fue una sorpresa que Android fuera capaz de manejar las secuencias rápidas de pulsado de botones y mover el personaje correctamente. Por el contrario, iOS es capaz de proporcionar una buena visión del juego, con objetos proporcionados, como coches, semáforos o el mismo personaje.

En conclusión, el proyecto ha alcanzado los objetivos marcados al comienzo satisfactoriamente. Evalúa una aplicación móvil multiplataforma en dos plataformas, Apple iOS y Android OS, concluyendo que Apple iOS ofrece una ligeramente mejor funcionamiento. La aplicación multiplataforma es un juego educacional, que además de ser un medio para el proyecto tiene su propio valor como herramienta educacional para niños.

Futuros análisis sobre aplicaciones móviles multiplataforma podrían ser útiles para completar las conclusiones de este trabajo. En cuanto al juego educacional, hay dos posibilidades para un futuro cercano. Una es crear otro juego, basado en el presente, donde el entorno sea ya de tres dimensiones. El usuario podría realizar más acciones que simplemente moverse e interactuar con los puntos de sincronización. La otra opción es añadir nuevos módulos o características a este juego, manteniéndolo lo más simple posible para el usuario.